Middle East Technical University | Department of Computer Engineering

CENG 351

Data Management and File Structures

Fall 2020

Programming Assignment 2

Due date: 17.01.2021, Sunday, 23:59

1 Introduction

Your CengTube is now growing. It has thousands of users and videos. You noticed that it is not as fast as the time when there were a few videos and users. Before it becomes a serious problem, you need to consider indexing.

In this assignment, to store the videos in our system, you are going to create a **primary B+ tree** on the vidID field i.e. the primary key field of the video records (described in detail later). You will implement only certain operations for the sake of simplicity.

2 Objectives

In summary, there will be 3 main operations to complete;

- Add Video: Adds a new video into tree
- Search Video: Searches for a video with the given vidID, prints the visited nodes along the path
- Print Tree: Prints the whole tree in depth first order, with proper indentation using tab characters

Your implementation is going to perform the specified operations based on the given input and will print the necessary output afterwards.

3 Project Structure

3.1 GUI Classes

The following classes are implemented to provide you a better environment for debugging. It is not expected for you to modify those.

- CengNodeType.java
- CengGUI.java
- GUIInternalNode.java
- GUILeafNode.java
- GUILevel.java
- GUITreeNode.java
- WrapLayout.java
- CengVideo.java: Defines the class for the video and its fields.
- CengVideoRunner.java: Includes the main method to run the application which has 3 arguments; order of the B+ Tree, path to the input file, enable GUI flag.

  Sample Compile and Run: (You can also use an IDE during development)

```
javac *.java
java CengVideoRunner 5 20InputSorted.txt True
```

3.2 Implementation

You will implement the following classes;

- CengTree.java: The three operations (add, search, print) that you need to implement are defined here.
- CengTreeNodeLeaf.java: Defines the class for the leaf nodes of the tree.
- CengTreeNodeInternal.java: Defines the class for the internal nodes of the tree.
- CengTreeNode.java: Parent class of leaf and internal nodes.
- CengTreeParser.java: To parse the input from `System.in` and input files.

There are *TODO* items inside those classes to direct you towards the implementation. Please read the explanations there.

4 Input/Output

The evaluation of the homework will be done without using the GUI. GUI is used to visualize the tree and call the necessary functions by clicking the buttons. There are 2 input files provided in order to allow GUI to load some examples. You can see how those are parsed when you open the GUI. Also, there is a video uploaded under the Assignment 2 section on our ODTUClass page, check it out to learn about how to use the GUI and how to run the application. Note that, the GUI shows the pointers between the parent-child nodes using colour notation and pointers between leaf nodes are out-of-scope of this homework.

You are going to construct the primary B+ Tree based on the numeric <vidID> field. The order d of the B+ tree will be given as an input. Note that, the order applies to leaf nodes as well, i.e., a leaf node will store N records, where $d \leq N \leq 2d$

You should continuously listen for the inputs from the command line in an infinite loop until you read the string "quit". When "quit" is read, just terminate the program. Here are some input & output examples.

4.1 Add

Input format: add|<vidID>|<videoTitle>|<channelName>|<category>

Example Input: add|3|Sad But True|Metallica|Music
Output: None

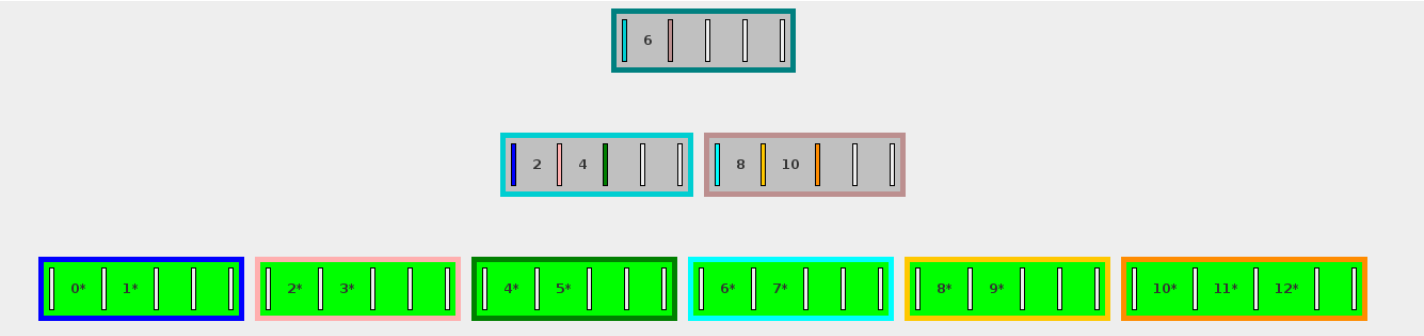
Note the use of the pipe character | and the lack of spaces between arguments, in order to allow spaces within each argument.

4.2 Print

Should be triggered when “print” is read as input.

Input format: print
Output format:
All non-leaf and leaf nodes should be printed with a proper indentation (with tabs) for each level in the tree.
For non-leaf nodes, you should print the search key enclosed by <index> and </index> tags.
For leaf nodes, you should print the content between <data> and </data> tags. Records should be printed between with <record> and </record> tags.

Current State of Tree (with the order $d = 2$):



Example Input: print

Example Output:

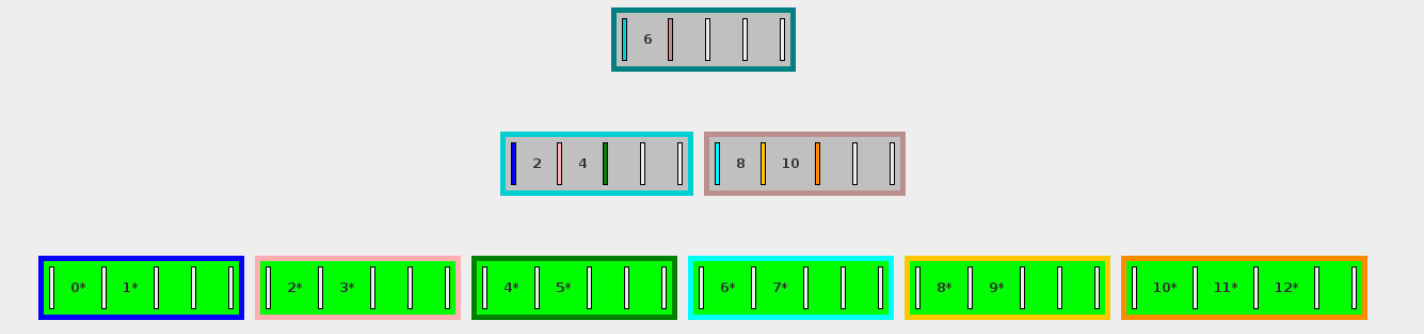
```
<index>
6
</index>
        <index>
        2
        4
        </index>
                <data>
                <record>0|Cheeseburger Recipe|Gordon Ramsay|Food</record>
                <record>1|Sad But True|Metallica|Music</record>
                </data>
                <data>
                <record>2|Playing Minecraft|Gamer Channel|Gaming</record>
                <record>3|News of Today|DW News|News</record>
                </data>
                <data>
                <record>4|Killa|140 Journos|Documentary</record>
                <record>5|Enter Sandman|Metallica|Music</record>
                </data>
        <index>
        8
        10
        </index>
                <data>
                <record>6|Fear Of The Dark|Iron Maiden|Music</record>
                <record>7|Freestyle Swimming|OlympicChannel|Sports</record>
                </data>
                <data>
                <record>8|Usain Bolt 100 meter|OlympicChannel|Sports</record>
                <record>9|Master Breathing|OlympicChannel|Sports</record>
                </data>
                <data>
                <record>10|New Camera|PewDiePie|Entertainment</record>
                <record>11|A Random Film|BestFilms|Films</record>
                <record>12|Java Tutorial|LearnCoding|Education</record>
                </data>
```

4.3 Search

Input format: search|<search_key>

Output: Prints the visited nodes starting from root to data records in leaf nodes, indentation with tabs should be added while traversing down on each level in the tree. If the record is not found, prints "Could not find <search_key>."

Current State of Tree (with the order $d = 2$):



Example Input: search|4
Example Output:

```
<index>
6
</index>
        <index>
        2
        4
        </index>
                <record>4|Killa|140 Journos|Documentary</record>
```

Example Input: search|29
Example Output: Could not find 29.

5 Submission

You are going to submit the files given under Section 3.2. Make sure you covered your implementation only under those files. Also, you should not use any subdirectories. Submission will be done via ODTUClass. Archive the files using `tar -cvf <e123456>.tar.gz`. Make sure that it gets extracted successfully by `tar -xvf <e123456>.tar.gz`

Before submitting, run your code without using any IDE with the following command;

```
javac *.java
java CengVideoRunner <order> <inputFileName> <enableGUIFlag>
```

6 Regulations

1. Programming Language: Java (Version13).
2. Attachments: Necessary source files are provided.
3. Cheating: We have zero tolerance policy for cheating. People involved in cheating will be punished according to the university regulations.